ACEEE

Int. J. on Recent Trends in Engineering and Technology, Vol. 10, No. 1, Jan 2014

Priority based K-Erlang Distribution Method in Cloud Computing

Chandan Banerjee[1, 2], Anirban Kundu[2, 3], Ayush Agarwal[1], Puja Singh[1], Sneha Bhattacharya[1] and Rana Dattagupta[4]

[1]Netaji Subhash Engineering College, Kolkata 700152, India

{chandanbanerjee1, ayush.agarwal2310, singh.puja.07.08, sneha.22.7}@gmail.com

[2]Innovation Research Lab (IRL), Howrah, West Bengal 711103, India

anik76in@gmail.com

[3]Kuang-Chi Institute of Advanced Technology, Shenzhen 518057, P.R.China

[4]Jadavpur University, Kolkata 700032, India

anirban.kundu@kuang-chi.org , rdattagupta@cse.jdvu.ac.in

***Abstract*— In this paper, we have proposed a priority based service time distribution method using Erlang Distribution for K-phases. The process of entering into the cloud is typically based on the priority of the service and form a queue. Each user needs to wait until the current user is being served if priority of the service is same, as per the First Come First Served policy. If the server is busy, then the user request has to be waited in the queue until the current user receives the result of the prescribed tasks. We have considered a priority class refers to a collection of all customers having the same priority. We introduce $M/E_K/1$ model for a priority based single server and $M/E_K/2$ model for priority based two servers in each class in cloud computing scenario to reduce overall mean queue length and waiting time. We have considered a single server and two server retrial queueing systems in which customers arrive in a Poisson process with arrival rate λ. We considered a single server and two server queues with three classes of customers.**

***Index Terms*— Cloud Computing, Erlang Distribution for K-phases, Waiting time, Queue length, Priority based service class.**

I. INTRODUCTION

Cloud computing is a new cost-efficient computing paradigm in which information and computer resources has been accessed from Web browser by users. Queueing systems in which arriving customers who find the server busy may retry for service after a period of time is called Retrial queues. In our system there are three priority classes, which define the order in which the service will be provided, since scheduling the queue in this manner may cause starvation, therefore we have implemented ageing technique to prevent this. If a new customer does not find any free server after connecting to the cloud service, then the system automatically redirects the request towards a waiting queue. At that moment, if the waiting queue is also fully occupied by other customers, then the newly arrived customer has to retry for service after certain time period. This technique is known as Retrial queues [1, 2]. Priority of service request is an important issue in scheduling because some request should be serviced earlier than others. These are requests which can't stay for a long time in a system. The $M/E_k/n$ (in this case, n=1 or n=2) queuing system with cloud computing service station is very useful to provide basic framework for efficient design and analysis of several practical situations including various technical systems. In cloud computing environment, multiple servers have been utilized

DOI: 01.IJRTET.10.1.1289

instead of single server to achieve higher computation power. In this manner, the system performance is increased reducing the mean queue length and waiting time. In multiple servers [3] scenario, cloud service [4] request needs no waiting time for a long time period and also the queue length would be less.
Our proposed approach using M/EK/1 [5, 6] and M/EK/2
[6, 7] priority based models has been described in Section
II. Experimental results using priority based Erlang distribution method has been shown in Section III. Conclusion is depicted in Section IV.

II. PROPOSED WORK

In this section, a priority based single service channel queuing system cloud model $M/E_K/1$ has been proposed.
Consider a single server retrial queueing system in which customers arrive in a Poisson process with arrival rate λ. These customers are identified as primary calls**.** Further assume that negative customers arrive at a rate **ν** which follows a Poisson process**.** Gelenbe [8] has introduced a new class of queueing processes in which customers are either Positive or Negative. Positive means a regular customer who is treated in the usual way by a server. Negative customers have the effect of deleting some customer in the queue. In the simplest version, a negative arrival removes an ordinary positive customer.
Let k be the number of phases in the service station. Assume that the service time has Erlang-k distribution [9] with service rate kμ for each phase. We assume that the services in all phases are independent and identical and only one customer at a time is in the service mechanism. If the server is free at the time of a primary call arrival, the arriving call begins to be served in Phase 1 immediately by the server then progresses through the remaining phases and must complete the last phase and leaves the system before the next customer enters the first phase. If the server is busy, then the arriving customer goes to orbit and becomes a source of repeated calls. This pool of sources of repeated calls may be viewed as a sort of queue. Every such source produces a Poisson process of repeated calls with intensity **σ**. If an incoming repeated call finds the server free, it is served in the same manner and leaves the system after service, while the source which produced this repeated call disappears. Otherwise, the system state does not change.
There are 'k' numbers of phases in our cloud system. Basic model of single server is shown in Fig. 1. Number of phases is four, if service is not available; where as if service is available, then number of phases becomes five. Only one server is considered in this case. We have considered three priority classes. The buffer is assumed to be infinite in each class. The priority based queuing discipline is first-come-first-serve (FCFS). Users' requests are being served one at a time. A new service with same priority could not be started until all the k-phases have been executed. Therefore, each arrival of users' requests increases the number of phases by k in the overall system. A mathematical basis for queuing theory [10] is provided for better understanding and for higher prediction based on the behavior of communication network.
We assume that the access from the queue to the service facility follows the exponential distribution which may depend on the current number n, $(n \geq 0)$ the number of customers in the orbit. That is, the probability of repeated
attempt during a particular interval , given that there are n customers in the orbit at time t is $n\sigma\,\Delta t$. It is called the classical retrial rate policy. The input flow of primary calls, interval between repetitions and service time in phases are mutually independent.

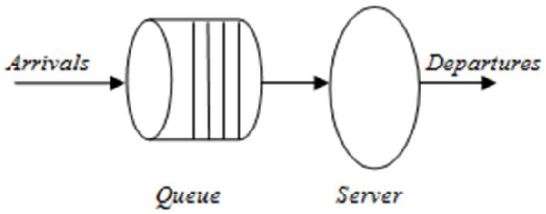

Fig.1. Basic Single Server Queuing Model

λ = Arrival rate of customers
μ = Expected number of customers completing service per unit time

Expected number of phases in the system =

$$Ls(k) = \frac{k+1}{2} * \frac{\lambda}{\mu-\lambda} \quad (1)$$

Expected number of phases in the queue =

$$Lq(k) = \frac{Ls(k)}{\mu} = \frac{k+1}{2} * \frac{\lambda}{\mu(\mu-\lambda)} \quad (2)$$

Expected number of customers in the queue =

$$Lq = \frac{Ls(k) - \text{average number of phases in the service}}{k}$$

$$= \frac{1}{k} * \left[\frac{k+1}{2} * \frac{\lambda}{\mu-\lambda} - \frac{k+1}{2} * \frac{\lambda}{\mu}\right]$$

$$= \frac{k+1}{2k} * \frac{\lambda^2}{\mu(\mu-\lambda)} \quad (3)$$

Since, $1/\mu$ is the average service time per customer and $(k+1)/2$ is the average number of phases of one customer in service; therefore time taken for serving a customer will be $(k+1)/2\mu$. Thus, average numbers of phases that arrive in this time would be $\lambda(k+1)/2\mu$.

$$\text{Average service time per customer} = \frac{1}{\mu} \quad (4)$$

$$\text{Average number of phases of one customer in service} = \frac{k+1}{2} \quad (5)$$

Expected waiting time of a customer in a queue =

$$Wq = \frac{Lq}{\lambda} = \frac{k+1}{2k} * \frac{\lambda}{\mu(\mu-\lambda)} \quad (6)$$

Expected waiting time of a customer in the system =

$$Ws = Wq + \frac{1}{\mu} \quad (7)$$

Expected number of customers in the system = $L_s = L_q + \frac{\lambda}{\mu}$

$$= \frac{k+1}{2k} \cdot \frac{\lambda^2}{\mu(\mu-\lambda)} + \frac{\lambda}{\mu}$$

$$= \lambda \left(\frac{k+1}{2k} \cdot \frac{\lambda}{\mu(\mu-\lambda)} + \frac{1}{\mu}\right)$$

$$= \lambda W_s \quad (8)$$

$$Average\ service\ time\ for\ k-phase = \frac{1}{k\mu} \quad (9)$$

Consider, $M/E_K/2$ queuing system having two servers (server 1 & server 2). Assume that the service time follows exponential distributions with the service rate '$\mu 1$' for "server 1" and '$\mu 2$' for "server 2"; such that $\mu 1 > \mu 2$. Customer arrival technique is considered as well-known "Poisson" process with rate λ and system has the provision for waiting queue. Each service request requires exactly one server for execution of the service; and the same priority service is served using typical FCFS technique. When there are several requests of same priority customers in the waiting queue, and a server is being released from its previous

task(s), then the first available customer request in the queue has been granted for execution. On the other hand, if a customer request arrives to an empty queue, it is directed to "server 1" immediately with a probability 'p' and the "server 2" with probability "1 – p". In case of two server based Erlang model, we have considered four numbers of phases, if service is not available at that moment. If service is available at any particular time, then the numbers of phases become six.

The priority of the requested service entering in the system is first checked. If the priority is 1, then that particular service has the highest priority and the process of the service starts. If the priority is 2 or 3 then the existence of a customer service with a higher priority is checked. However, in our system, we have provided only three different priority classes for convenience. If there exist a service request with a higher priority in the queue then that service request (with the lower priority) has to wait till all the higher priority service request get the service. The priority of service classes 2 and 3 increases by one after every 60 seconds. This helps to avoid starvation to a great extent. After this, the requested service is served as per Erlang's k-phase distribution with two servers.

The flowchart of priority based different 'k' phases in proposed cloud for two servers has been shown in Fig. 2.

Expected number of customers (not phases) in the queue: $$Lq = \frac{(K+1)*\lambda^3}{2K\mu(4\mu^2-\lambda^2)} \quad (10)$$

It has been observed using equations (3) and (10) that difference between one server based 'L_q' is $(\mu-\lambda)$, and difference between two server based 'L_q' is $(4\mu^2-\lambda^2)$. Mathematically, $(4\mu^2-\lambda^2)$ is always greater than $(\mu-\lambda)$. Therefore, value of expected number of customers in the queue for two servers is always less than the value of expected number of customers in the queue for single server. Waiting time using two servers based queuing method is also decreased.

Expected number of customers in the system: $$Ls = Lq + \frac{\lambda}{\mu} \quad (11)$$

Expected waiting time of a customer in a queue:

$$W_q = \frac{L_q}{\lambda} \quad (12)$$

Expected waiting time of a customer in the system: $$Ws = Wq + \frac{1}{\mu} \quad (13)$$

t_{av} = Average Waiting Time

Here, we have considered a priority class refers to, a collection of all customers having the same priority. For example when the net arrival rate of a customer is say 20, we assume the arrival rates for the first, second and third priority classes are respectively. We have assumed that a priority class with lesser value of i has more priority and less arrival rate, i.e., the arrival rate of a class varies inversely as the priority of that class.

W_n = Steady state waiting time for priority-n class.

$$W_n = \frac{1}{AB_{n-1}B_n} \quad (14)$$

$$A = s!\frac{s\mu-\lambda}{r^3}\sum_{j=0}^{s-1}\frac{r^j}{j!} + s\mu \quad (15)$$

$$B_n = 1 - \frac{\sum_{i=1}^{n}\lambda_i}{s\mu} \quad (16)$$

S = Number of servers.

μ = Mean service rate per busy server.

λ_i = Mean arrival rate for priority class i.

$$\lambda = \sum_{i=1}^{n}\lambda_i$$

$$r = \frac{\lambda}{\mu}$$

We have assumed that 1 has been considered as highest priority and we also assume that the arrival rates of the priority classes vary exponentially.

For example, for the ith priority class, the arrival rate

$\lambda_i = (x_1)^i$,

Where x_1= arrival rate of the 1st priority class and so on.

The users' service requests (UR) is provide as input to scheduler server. Then it gets the total available resources (AR) and the number of running VMs in the data center in cloud. Find a list of appropriate VMs (AVM) capable of provisioning the requested service class. Algorithm 1 described the priority based service request handling in each class and Algorithm 2 shows for customer request with priority handling using different phases in proposed Cloud.

In general, in Erlang's k-phase distribution, the expected waiting time for a customer I the system,

$$Ws = \frac{k+1}{2k} * W_q + \frac{1}{\mu} \qquad (17)$$

Now, for a priority class-i, we have,

$$Ws_i = \frac{k+1}{2k} * (1/\ AB_{i-1}B_i) + \frac{1}{\mu} \qquad (18)$$

In this section some important performance measures along with formulas and their qualitative behaviour for various values of λ, μ, k , ν and σ are defining by P(n , 0) = Probability that there are n customers in the orbit and server is free

P(n , i) = Probability that there are n customers in the orbit and server is busy with customer in the ith phase

The probability mass function of Server state Let S(t) be the random variable which represents the phase in which customer is getting service at time t .

P: $\sum_{i=0}^{\infty} p(i,0)$ where s : 0

In this way,

P: $\sum_{i=0}^{\infty} p(i,k)$ where s : k

The Mean number of busy servers

$$MNBS = \sum_{i=0}^{\infty} \sum_{j=1}^{k} p(i,j) \qquad (19)$$

The probability mass function number of customers in the orbit

Let X(t) be the random variable representing the number of customers in the orbit.

Prob (No customers in the orbit) = $\sum_{j=0}^{k} p(0,j)$

Prob (i customers in the orbit) = $\sum_{j=0}^{k} p(i,j)$

The Mean number of customers in the orbit

$$MNCO = \sum_{i=0}^{\infty} i \left(\sum_{j=0}^{k} p(i,j)\right) \qquad (20)$$

The probability that the orbiting customer is blocked

Blocking Probability =

$$\sum_{i=1}^{\infty} \sum_{j=1}^{k} p(i,j) \qquad (21)$$

The probability that an arriving customer enter into service immediately

$$PSI = \sum_{i=0}^{\infty} p(i,0) \qquad (22)$$

Algorithm 1:

Priority Based Service Request Handling in each class (PBSRHC)

Input: User Service Request with priority
Output: Select a Resource

Step 1. Form Available VM List and
UsedVMList and also available ResourceList from each priority class

Step 2. If AVM (UR,AR) != ϕ then
VM=load-balancer (AVM (UR,AR)) and
Deploy service on VM and
deploy=true
Else if UR has advance reservation
Deployed=true
Else if Resource Able To Host ExtraVM then
Start newVMInstance
Add VMToAvailbaleVMList
Deploy service on newVM
Deployed=true
Else
queue serviceReuest until queueTime > waitingTime
Deployed=false
End if
Step 3. If deployed then
return successful
stop
Else
return failure
stop

Algorithm 2:

Algorithm for customer request with priority handling using different phases in proposed Cloud.

Input: A service request by client in a cloud.
Output: Requested service allocated to a particular server.

Step 1: Authentication of users
Step 2: Check Priority class. If priority ==1 then goto step 3 else if priority ==2 then goto step 4 else goto step 5.
Step 3: If server busy, wait for completion of earlier priority 1 request otherwise allocate request to the server and provide service.
Step 4- (a): Wait for completion of priority 1 customer request or wait for completion of earlier same priority request; set timer =1 and timer = timer+1 continues until timer > 60.
Step 4-(b): If timer > 60 then priority = Priority - 1 and goto step 3.
Step 5- (a): Wait for completion of priority 1 and priority 2 customer requests or wait for completion of earlier same priority request; set timer =1 and timer = timer+1 continues until timer > 60.
Step 4-(b): If timer > 60 then priority = Priority - 1 and goto step 4(a).

III. Experimental Results

We have evaluated the performance of our priority based K-phase Erlang distribution method through simulations. With different set of service request simulation is done. In each run of simulation, consider a single server retrial queuing priority based system with Erlang-k distribution. Expected waiting time of a customer in the system is increased based on the increment in case of arrival rate in single server as referred

in Figure 2, Figure 5, Figure 8 and Figure 9 and multiple server scenarios as referred in Figure 3, Figure 4, Figure 6 and Figure 7.

For λ=20, we have

λ_1=2.31 (for priority class 1)

λ_2=5.34 (for priority class 2)

λ_3=12.32 (for priority class 3)

For λ=60 , we have

λ_1=3.53 (for priority class 1)

λ_2=12.46 (for priority class 2)

λ_3=43.98 (for priority class 3)

For λ=120 , we have

λ_1= 4.56 (for priority class 1 – P1)

λ_2=20.79 (for priority class 2 – P2)

λ_3=94.81 (for priority class 3 – P3)

We see that the highest priority class has the lower arrival rate.

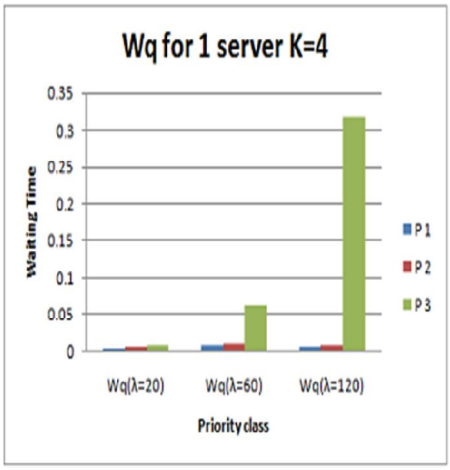

Fig 2. Wq for M/E_k/1 with K=4

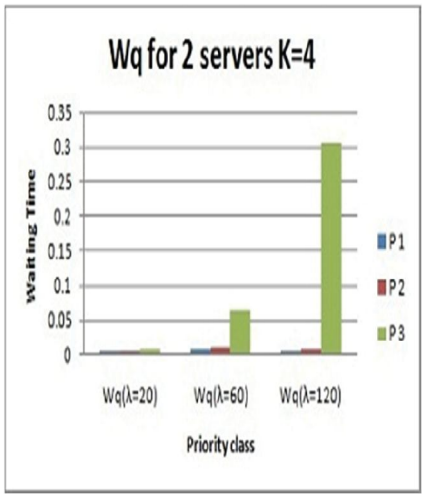

Fig 3. Wq for M/E_k/2 with K=4

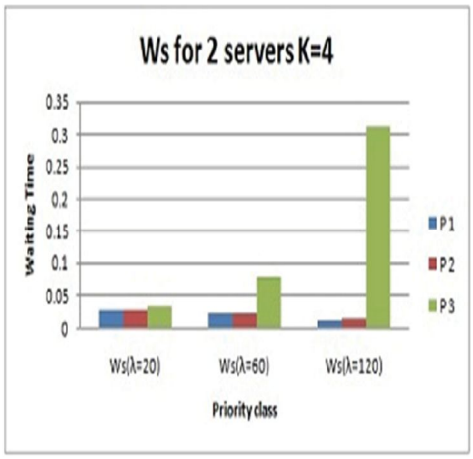

Fig 4. Ws for $M/E_k/2$ with K=4

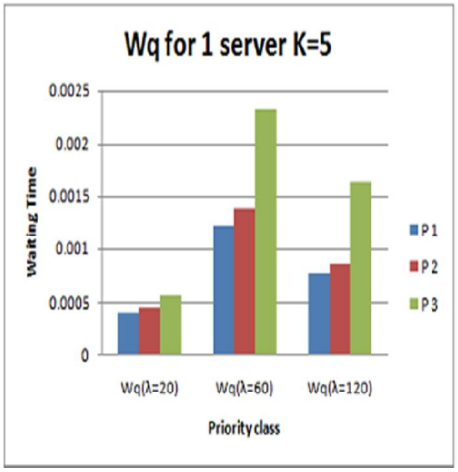

Fig 5. Wq for $M/E_k/1$ with K=5

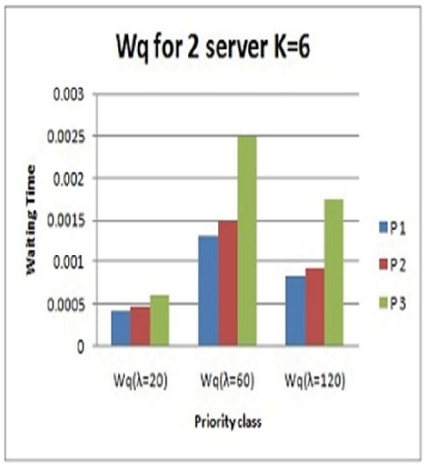

Fig 6. Wq for $M/E_k/2$ with K=6

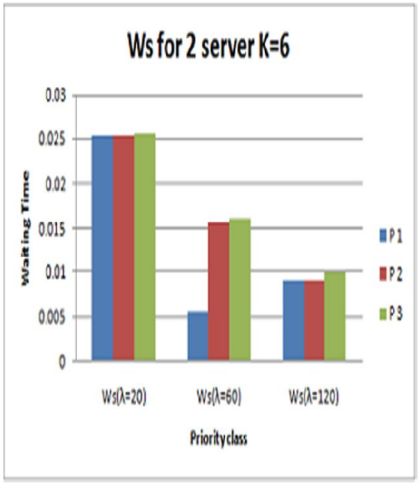

Fig 7. Ws for $M/E_k/2$ with K=6

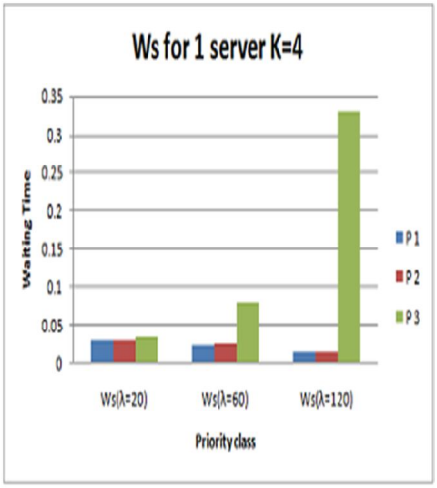

Fig 8. Ws for $M/E_k/1$ with K=4

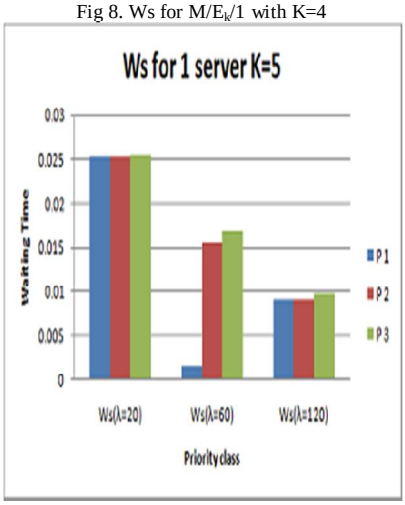

Fig 9. Ws for $M/E_k/1$ with K=5

IV. CONCLUSION

In this paper, we have successfully designed a queue based model for priority based Erlang Distribution for k- phases. Our design consists of single server with 4 and 5 phases and two servers with 4 and 6 phases. We have considered three priority classes. In this model, service requests have been executed using queues. Virtual machines [11] are modeled as service centers using $M/E_K/1$ model and $M/E_K/2$ model. Multiple servers based scenario has improved the performance based on our system based on queue length reduction and waiting time optimization over single server based scenario. Numerical results have been demonstrated exhibiting higher performance in case of our priority based approach for $M/E_K/2$ which reduces the queue length and waiting time compared to $M/E_K/1$. The low priority of service classes 2 and 3 increases by 1 after every 60 seconds. This approach avoids starvation. $M/E_K/1$ has produced better result compared to M/M/1 [6, 12]. $M/E_K/2$ has shown better throughput compared to M/M/2 [6, 12] in terms of average queue length and average waiting time.

REFERENCES

[1] J. R. Artalejo, "A queueing system with returning customers and waiting line," Operations Research Letters, Vol. 17, pp. 191-199, 1995

[2] G. I. Falin, "A survey of retrial queues," Queueing Systems, Vol. 7, 127-167 (1990)

[3] F. Ayad, M. Barsoum, A. Hasan, "On Verifying Dynamic Multiple Data Copies over Cloud Servers," IACR Cryptology ePrint Archive 2011:447 (2011)

[4] Anirban Kundu, Chandan Banerjee, Priya Saha, "Introducing New Services in Cloud Computing Environment," International Journal of Digital Content Technology and its Application, Vol. 4, No. 5, (2010)

[5] *M. Jain, P. K. Agrawal, "$M/E_k/1$ Queueing System with Working Vacation,"* ICAQM, Vol. 4, No. 4, pp. 455-470 (2007)

[6] Chandan Banerjee, Anirban Kundu, Ayush Agarwal, Puja Singh, Sneha Bhattacharya, Rana Dattagupta; "K-phase Erlang Distribution method in Cloud Computing"; Fourth International Conference on Advances in Communication Network and Computing – CNC 2013; LNICST pp. 53~59.

[7] J. R. Murray, W. D. Kelton, "The Transient Response of the $M/E_K/2$ Queue and Steady-State Simulation," Technical report University of Michigan, 85-29, pp. 1-20 (1985).

[8] E. Gelenbe "Product-Form queueing networks with negative and positive customers", Journal of Applied Probability, Vol. 28 (3): 656–663 (Sep., 1991).

[9] Donald Gross and Carl M. Harris, Fundamentals of Queueing theory, (1974)

[10] H. Xiao, G. Zhang, "The Queuing Theory Application in Bank Service Optimization," International Conference on Logistics Systems and Intelligent Management, IEEE, **Vol.** 2, **pp.** 1097 – 1100 (2010)

[11] M. H. Jamal, A. Qadeer, W. Mahmood, A. Waheed, J. J. Ding, "Virtual Machine Scalability on Multi-Core Processors Based Servers for Cloud Computing Workloads," IEEE International Conference on Networking, Architecture, and Storage, 2009, pp. 90-97 (2009)

[12] S. Sowjanya, D. Praveen, K. Satish, A. Rahiman, "The Queueing Theory in Cloud Computing to Reduce the Waiting Time," IJCSET, Vol. 1, Issue 3, pp. 110-112 (2011)